Lezione 11-12

March 7, 2012

Comandi: Formalizzazione, Uso, Implementazione

- Uso: dove e perchè
- Comandi Strutturati e non
- Comandi per il Controllo: Semantica e Implementazione
- Blocchi inline: Semantica e Implementazione
- Astrazioni di controllo: Uso, Semantica, Implementazione

Comandi: Generalità e Uso

- Presenti solo nei Linguaggi Procedurali e loro estensioni (anche OO: C#, Java..)
  - Coinvolgono e sono basati su operazioni di modifica dello stato (assegnamento,...)
  - Composizione di base: Sequenza
  - Altre forme di Composizione: definite da Comandi per il Controllo e la Modifica della Sequenza (di modifiche dello stato)
  - ... e conducono alla **Prescrittività** del linguaggio e dell'algoritmo di calcolo

Comandi per il Controllo

- Comandi per il Controllo e la Modifica della Sequenza (di modifiche dello stato)
  - **Esplicito**. `goto`, `break`, `continue`, `return`
  - **Condizionato**. `if`, `case`, `switch`, `cond`
  - **Iterativo**. `determinato`, `indeterminato`
  - (**Ricorsione**). Presente per astrazioni induttive orientato al calcolo piuttosto che alla modifica della sequenza.

Forma dei costrutti: Comandi Strutturati

- Comandi per il Controllo e la Modifica della Sequenza (di modifiche dello stato)
- Sono composti di altre strutture
  - più semplici
  - espressioni e/o comandi
  - Comando Strutturato: ha un unico punto di ingresso ed un unico punto di uscita per il flusso di controllo di sequenza
  - Ingresso e uscita coincidono con ingresso e uscita del codice generato dal compilatore per la loro traduzione

Example

Un comando blocco, contenente un comando strutturato:
`{...while E do {c1;...;ck};...}`

Forma dei costrutti: Comandi non strutturati

- Comando non Strutturato

Example

Un comando blocco, contenente vari comandi alcuni strutturati altri no

```
{ ...A:while E do {c1;...;ck};...; ...goto A;...}
```

L'uscita non coincide con l'uscita del codice generato dal compilatore.

- Tutti i comandi di trasferimento esplicito sono non strutturati
- Invocazioni di procedura, Eccezioni, ... (e altri ancora) richiedono l'uso di comandi non strutturati

Combinare Comandi strutturati e non

- La presenza di comandi non strutturati può condizionare il comportamento di quelli strutturati

Example

Un comando blocco, contenente vari comandi alcuni strutturati altri no:
`{...while...do {...A: ...};...;goto A;...}`

- Costruzioni con comportamenti difficile da definire (formalmente)
- ... conducono a costruzioni difficili da usare (in modo consapevole)
- Limitarne la composizione
- ... anche se ciò contrasta con il principio di **ortogonalità** di un linguaggio

Comandi: Semantica/1

Table10 – Semantica dei Comandi – 1.

Domini Sintattici

$$
\begin{aligned}
\texttt{C} ::= \;& \texttt{C C} \mid \texttt{E} = \texttt{E} \\
& \mid \texttt{IF E Then C Else C} \mid \texttt{Case E S} \\
& \mid \texttt{For I} = \texttt{E To E By E Do C} \\
& \mid \texttt{While E Do C} \\
& \mid \{\texttt{D C}\} \\
& \mid \texttt{Proc I()C} \mid \texttt{Call I()}
\end{aligned}
$$

Assegnamento nei linguaggi/1

Example

L'assegnamento in C è sia un operatore su epressioni sia un comando (due costrutti distinti: stessa sintassi concreta, ma diversa sintassi astratta e semantica).
L'assegnamento in Java é solo un operatore su espressioni. Però le espressione Java sono comandi e possono essere composti anche con la sequenzializzazione.

Comandi: Semantica/2

Funzioni Semantiche

$\mathcal{C}[\![\mathtt{C}]\!]_\rho : \mathtt{Store} \to \mathtt{Store}_\perp$ (*Comandi*)

$\mathcal{M}[\![\mathtt{C}_1\ \mathtt{C}_2]\!]_\rho = \mathcal{M}[\![\mathtt{C}_1]\!]_\rho \circ \mathcal{M}[\![\mathtt{C}_2]\!]_\rho$

$\mathcal{M}[\![\mathtt{E}_l = \mathtt{E}_r]\!]_\rho =$

$\lambda s.\ \mathtt{Let}\{(v_1, s_1) = \mathcal{E}[\![\mathtt{E}_r]\!]_\rho(s)\}$

$\{(l_2, s_2) = \mathcal{E}[\![\mathtt{E}_l]\!]_\rho(s_1)\}\mathtt{upd}(l_2, \mathtt{VM}(v_1), s_2)$

Example

Ricordiamoci la differenza tra sintassi concreta e sintassi astratta. Il comando di assegnamento nel frammento sotto (espresso con l'usuale sintassi concreta, ad esempio, del linguaggio C):

```
x = y + 5 * x
```

è di fatto il termine seguente (quando espresso nella sintassi astratta dal front-end di interprete o compilatore:

```
Den(x)= Val(y) + 5 * Val(x)
```

Assegnamento nei linguaggi/2

Example

L'assegnamento definito dalla semantica precedente valuta gli operandi in un ordine particolare: Prima l'operando del valore memorizzabile, dopo l'operando dell'espressione denotabile. Questo infatti, il comportamento dell'assegnamento in AlgolW e in Pascal. Il linguaggio C e successivamente Java, valuta gli operandi dell'assegnamento da sinistra a destra, come per un qualunque operatore.

(a) Interessante, scrivere un programma contenente un frammento di codice in Pascal e in C che mostri i differenti comportamenti;

(b) Interessante, scrivere la semantica dell'assegnamento per C/Java

Comandi: Semantica/3

Funzioni Semantiche

$$\mathcal{M}[\![\texttt{IF E Then } C_1 \texttt{ Else } C_2]\!]_\rho = \ldots \text{ (\textit{vedi Esercizio}...)}$$
$$\mathcal{M}[\![\texttt{Case E S}]\!]_\rho = \ldots \quad \text{(\textit{vedi Esercizio}...)}$$

Comandi: Semantica/4

Funzioni Semantiche

$$\mathcal{M}[\![\texttt{For I} = \texttt{E}_1 \texttt{ To E}_2 \texttt{ By E}_3 \texttt{ Do C}]\!]_\rho(\texttt{s}) =$$
$$\texttt{Let}\{(\mathit{inizio}, \texttt{s}_1) = \mathcal{E}[\![\texttt{E}_1]\!]_\rho(\texttt{s})\}$$
$$\{(\mathit{fine}, \texttt{s}_2) = \mathcal{E}[\![\texttt{E}_2]\!]_\rho(\texttt{s}_1)\}$$
$$\{(\mathit{passo}, \texttt{s}_3) = \mathcal{E}[\![\texttt{E}_3]\!]_\rho(\texttt{s}_2)\}$$
$$\{n = (\mathit{fine} - \mathit{inizio} + \mathit{passo})/\mathit{passo}, f = \mathcal{M}[\![\texttt{C}]\!]_\rho\}$$
$$f^n(\texttt{s}_3)$$

Comandi: Implementazione/1

- Decomposizione (compile time) di comandi in: sequenze di operazioni elementari
  - operazione di modifica di stato;
  - operazione, condizionata o no, di modifica del *Location/Program Counter*;

Comandi: Blocchi inline

Funzioni Semantiche

$$\mathcal{M}[\![\{\texttt{D C}\}]\!]_\rho(s) = \texttt{Let}\{(\rho_1, s_1) = \mathcal{D}[\![\texttt{D}]\!]_\rho(s)\}\ \mathcal{M}[\![\texttt{C}]\!]_{\rho_1}(s_1)$$

Example

Questi frammenti ora hanno un preciso significato:
{Var x = 5; Var y; y=x+2; {Var y = x; x=y=x;}; y=x;...}

- Implementazione:
  - AR: per blocchi inline
  - Stack di AR: per astrazioni di controllo ricorsive

Comandi: Semantica/4

Funzioni Semantiche

$$\mathcal{M}[\![\texttt{While E Do C}]\!]_\rho(\texttt{s}) = \mathtt{Y}f\,.\,\lambda\texttt{s}\,.\,\texttt{Let}\{(v,\texttt{s}_1) = \mathcal{E}[\![\texttt{E}]\!]_\rho(\texttt{s})\}\ \texttt{if}(\texttt{true}(v),\texttt{s}_1,(\mathcal{M}[\![\texttt{C}]\!]_\rho \circ f)(\texttt{s}_1))$$

Funzioni Ausiliarie

$\texttt{DV} : \texttt{Den} \longrightarrow \texttt{Val}$

$\texttt{VM} : \texttt{Val} \longrightarrow \texttt{Mem}$

$\texttt{true} : \texttt{Val} \longrightarrow \texttt{Boolean}$

Astrazioni di Controllo: Uso

- Uso
  - **Localizzazione** del codice di una funzionalitá dell'algoritmo implementato;
  - **Modifica** e **Certificazione** di una unità di programmazione;
  - **Riuso** del codice di una unità di P.;
  - **Ricorsione.** Meccanismo per calcoli definiti in modo induttivo;
  - **Valori Funzionali.** Programmazione Higher Order

Astrazioni di Controllo: Astrazioni Funzionali/1

Table11 – Semantica delle Espressioni 2

Domini Sintattici

$$
\begin{array}{lll}
\texttt{D} ::= \ldots & | \ \texttt{Function I(P}_1\ \texttt{I}_1\ \ldots\ \texttt{P}_n\ \texttt{I}_n\texttt{) E} & | \ \ldots \\
\texttt{E} ::= \ldots & | \ \texttt{A} & (\textit{Parametri Attuali}) \\
 & | \ \texttt{Call I(A}_1\ \ldots\ \texttt{A}_n\texttt{)} & (\textit{Invocazione}) \\
 & | \ \ldots &
\end{array}
$$

Astrazioni di Controllo: Astrazioni Funzionali/2

Funzioni Semantiche

$$\mathcal{D}[\![\texttt{D}]\!]_\rho : \texttt{Store} \to (\texttt{Env} \times \texttt{Store}_\perp)$$

$$\mathcal{D}[\![\texttt{Function I() E}]\!]_\rho(s) = (\texttt{bind}(\texttt{I}, \texttt{F}(\mathcal{E}[\![\texttt{E}]\!]_\rho), \rho), s)$$

Funzioni Ausiliarie

$$\texttt{F} : (\texttt{State} \to (\texttt{Val} \times \texttt{State})) \to \texttt{ProcFun}$$

Example

Il valore denotabile generato dalla seguente dichiarazione:

`Function h() x+y`

considerata in un ambiente ρ è `F(v)`, dove `v` è il seguente valore:

$$\lambda\ \texttt{s}.\ +_{\perp_s}(\texttt{MV}(\texttt{s}(\rho(x))), \texttt{MV}(\texttt{s}(\rho(y))))$$

Astrazioni di Controllo: Astrazioni Funzionali/3

Funzioni Semantiche

$\mathcal{E}[\![\mathtt{E}]\!]_\rho : \mathtt{Store} \to (\mathtt{Val}_\perp \times \mathtt{Store}_\perp)$

$$\mathcal{E}[\![\mathtt{Call\ I()}]\!]_\rho(s) = \mathtt{Let}\{\mathtt{F}(f) = \rho(\mathtt{I})\}\ f(s)$$

Domini Sintattici Ausiliari

....

Example

Se invochiamo `Call h()` in un ambiente δ con $\delta(h) = \rho(h)$ e una memoria $\mathtt{s}_1$ con ... allora il termine `f(s)` vale:

$$\lambda\ \mathtt{s}.\ +_{\perp_S}(\mathtt{MV}(\mathtt{s}(\rho(x))), \mathtt{MV}(\mathtt{s}(\rho(y))))(\mathtt{s}_1)$$

e si riduce a $+_{\perp_S}(3, 5)$, calcolando infine 8.

Sostituiamo i puntini delle premesse con quello che deve essere detto.

Funzioni definite ricorsive/2

Example

La dichiarazione di funzione, `Function I() E`, di cui abbiamo data la semantica non può essere definita ricorsiva.
(a) Interessante è applicare la definizione data ad una funzione definita ricorsiva e descrivere, dettagliatamente, cosa calcoli $\mathcal{E}[\![\mathtt{E}]\!]_\rho$;
(b) Interessante è modificare la definizione della funzione semantica $\mathcal{E}$ per definire la semantica per il pi generale caso di funzioni definite ricorsive.